Linkage Pattern Mining Method for Multiple Sequential Data with Noise

Saerom Lee, Takahiro Miura, Yusuke Okubo, and Yoshifumi Okada

***Abstract*—Linkage pattern mining is a data mining technique that finds frequent patterns that appear repeatedly across multiple sequential data. This technique does not assume similarity or correlation between the frequent patterns in a linkage pattern; thus, it is expected to be a promising approach for discovering causal association among events in multiple sensor data, such as physiological signals in different regions and crustal movements at different points. However, existing methods have focused only on detecting linkage patterns without noise/fluctuations in sequential data. This study's objective is to develop a new noise-robust linkage pattern mining method. The proposed method excludes pseudo patterns derived from noise using closed itemset mining from interval graphs regarding frequent patterns such that only noiseless and maximal linkage patterns are extracted. The proposed method is applied to artificial sequential datasets with embedded linkage patterns. Experimental results show that this method can adequately detect embedded linkage patterns without noise and previously undetectable embedded linkage patterns with noise.**

***Index Terms*—closed itemset, interval graph, linkage pattern, sequential pattern mining**

I. Introduction

Sequential pattern mining is a promising and effective data mining method for finding frequent patterns in large-scale sequential data. After Agrawal et al. [1] constructed the foundations of sequential pattern mining in 1995, various new effective algorithms have been developed [2], [3] and applied in a wide range of fields such as web log analysis [4], market basket analysis [5], behavior analysis [6], process analysis [7], and DNA sequence analysis [8]. Research into sequential pattern mining can be broadly classified into two types: approaches that target single sequential data and those that target multiple sequential data. The former aims to find repeating and frequently occurring patterns (frequent patterns or episodes) in sequential data [9]–[13]. The latter focuses on detecting same or similar subsequences among sequential data [14]–[16].

Manuscript received February 24, 2015. L. Saerom is with the Division of Production and Information Systems Engineering, Muroran Institute of Technology, 27-1, Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan (e-mail: saerom@cbrl.csse.muroran-it.ac.jp). T. Miura is with the IT Platform Division Group, Information & Telecommunication Systems Company, Hitachi, Ltd., 292, Yoshida-cho, Totsuka-ku, Yokohama, Kanagawa 244-0817, Japan (e-mail: takahiro.miura.mw@hitachi.com). Y. Okubo is with the Division of Information and Electronic Engineering, Muroran Institute of Technology, 27-1, Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan (e-mail: ookubo@cbrl.csse.muroran-it.ac.jp). Y. Okada is with the College of Information and Systems, Muroran Institute of Technology, 27-1, Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan (corresponding author to provide phone: +81-143-5408; fax: +81-143-5408; e-mail: okada@csse.muroran-it.ac.jp)

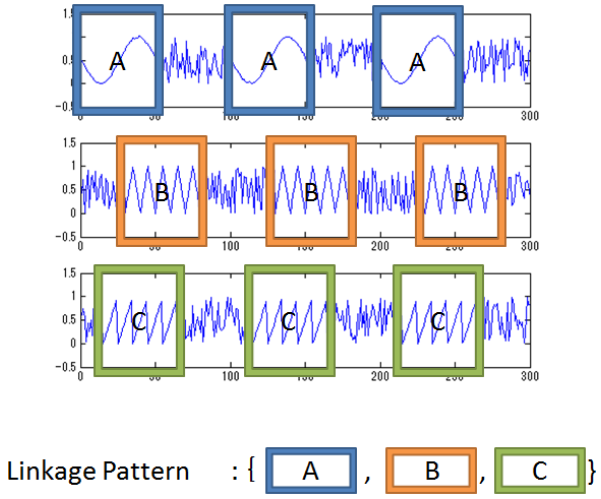

Fig. 1. Linkage pattern repeating across three sequential data

Recently, Miura and Okada [17] proposed a method for mining a linkage pattern that is a set of patterns that repeats across multiple sequential data. In their method, linkage patterns were extracted using an interval graph representation of frequent patterns in the sequential data. Note that linkage pattern mining does not assume similarity or correlation among different sequential data patterns. Fig. 1 shows an example of a linkage pattern {A, B, C} that appears across three sets of sequential data. As we can see, even if patterns that occur frequently in the respective sequential data do not show similarity to each other, the set of those patterns is extracted as a linkage pattern if it appears continually within the same period. Miura's method demonstrated good performance on sequential data without noise/fluctuations [17]; however, they suggested that noise/fluctuations within the sequential data can significantly affect the accuracy of extracting linkage patterns.

This study develops a noise-robust linkage pattern mining method by improving Miura's method. In our method, closed itemset mining is employed to exclude pseudo patterns generated by noise/fluctuations and obtain only frequent and maximal patterns among different interval graphs. In this study, comparative performance results between the proposed method and Miura's method (hereafter referred to as "the previous method") are shown using artificial sequential datasets.

The remainder of this paper is organized as follows. Section II defines linkage pattern. Section III defines closed itemsets. Section IV discusses problems with the previous

(Revised online publication: 22 July 2016)

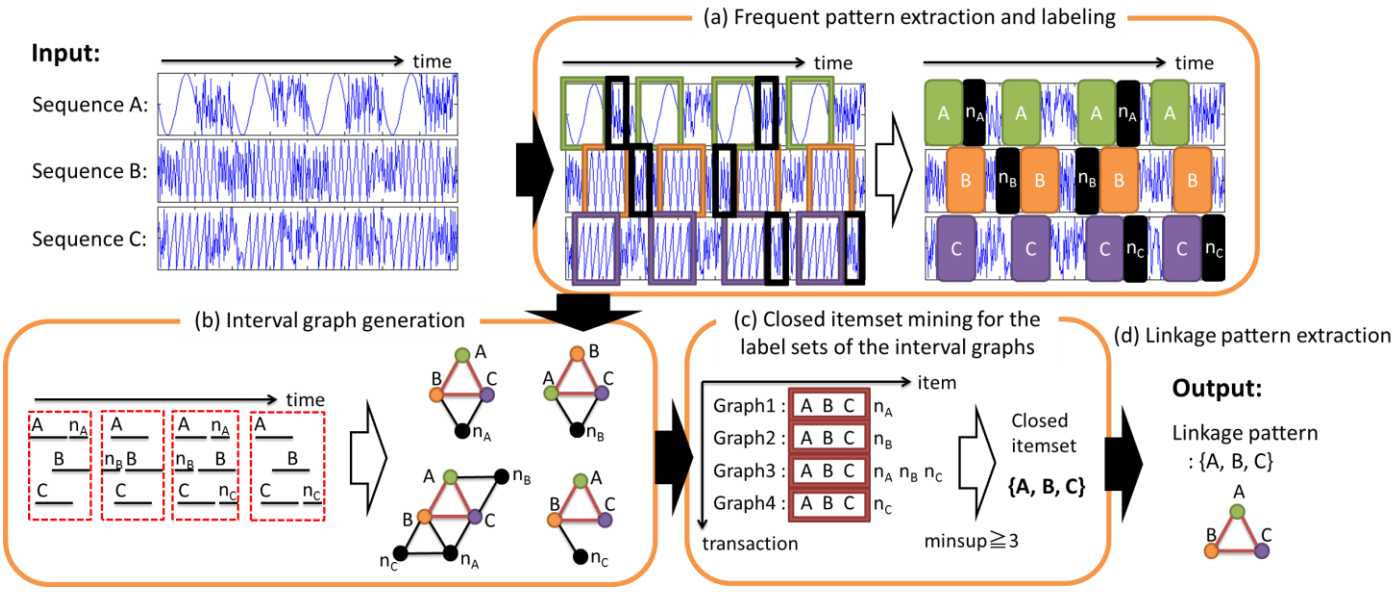

Fig. 2. Procedure of the proposed method

method and the procedure of the proposed method. Section V explains the experimental methods for performance evaluation using artificial sequential datasets. Section VI presents the experimental results and discusses some observations. Section VII provides an overall summary.

Note that this paper is an extended and revised version of our previous paper presented at IMECS 2014 [18].

II. Definition of Linkage Pattern

Let S be a single sequential data. $freq(S, \alpha)$ is the number of occurrences of a subsequence α in S. For a pre-defined constant value θ, α is a frequent pattern in S if $freq(S, \alpha) \geq \theta$.

Suppose that multiple sequential data are given as input, and that frequent patterns have already been extracted from those sequential data. If frequent patterns occurring over those sequential data in a certain time frame satisfy the following two conditions, the set of those frequent patterns is called a linkage pattern.

1) For all the frequent patterns, there exist one or more frequent patterns whose occurring time zones overlap partially or entirely with each other.
2) A set of the frequent patterns that satisfy condition 1) occurs x or more times along the sequential data.

III. Definition of Closed Itemset

Let $I = \{1, 2, \ldots, n\}$ be a set of *items*. A *transaction database* on I is a set $T = \{t_1, t_2, \ldots, t_m\}$ such that each t_i is included in I. Each t_i is called a *transaction*. A set $P \subseteq I$ is called an *itemset*. A transaction including P is called an *occurrence* of P. The set of occurrences of P is expressed as $T(P)$. The size of a set of occurrences for P is referred to as the *frequency* of P.

An itemset P is called a *closed itemset* if no other itemset Q satisfies $T(P) = T(Q)$, $P \subseteq Q$. For a given *minimum support* constant (hereafter *minsup*), P is frequent if $|T(P)| \geq minsup$. A frequent and closed itemset is referred to as a *frequent closed itemset*.

IV. Method

Fig. 2 shows the procedure of the proposed method. Fig. 2a, 2b, and 2d are the steps implemented in the previous method: extracting and labeling frequent patterns from each sequence (Fig. 2a), generating interval graphs depending on overlapping labels on the time axis (Fig. 2b), and outputting the linkage pattern (Fig. 2d). In this method, a new step (Fig. 2c) is introduced, i.e., closed itemset mining from the generated interval graphs. This resolves the problem by which linkage patterns are contaminated by noise data, as observed in the previous method. These steps are explained in detail below.

A. Frequent pattern extraction and labeling

First, normalization and discretization are executed on all sequential data in a preprocessing. In normalization, sequential data are converted to a scale from 0 to 1. In the discretization, the range of normalized data (0–1) is divided at the D stages, and a discrete value from 0 to D-1 is allocated to each data.

Next, repeatedly occurring frequent patterns are extracted from the sequential data using Mannila's algorithm [13]. This algorithm uses a window width w and a minimum number of occurrences θ as input parameters, where w and θ are natural numbers ≥ 2. Window width w is the length of the slice used to scan sequential data. The minimum number of occurrences θ is the minimum number of frequent patterns to be extracted. Mannila's algorithm finds frequent patterns that satisfy θ for a specified w.

The labeling process applies the same label to the same frequent pattern. This process is performed after excluding frequent patterns with length less than $w/2$. When multiple

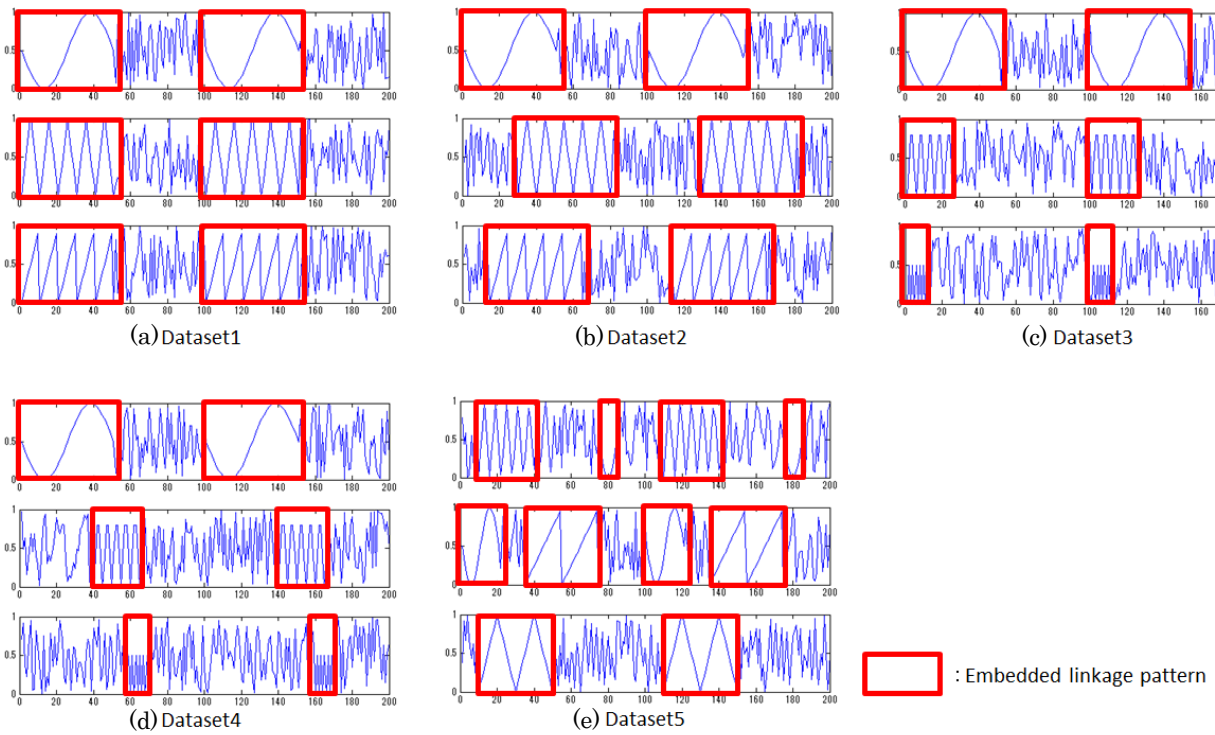

Fig. 3. Artificial datasets

frequent patterns occur within the same periods in the same sequential data, labeling is performed for the maximum length frequent pattern.

B. *Interval graph generation*

Here, a labeled frequent pattern is referred to as a *label*. In this step, interval graphs are generated from the interval representation of each label. An interval graph is obtained by associating each label with a node and an overlap of any two labels on the time axis between sequential data with an edge [19]–[21]. In other words, an interval graph is a set of frequent patterns that occur in a linked manner in the same period between different sequential data.

The previous method outputs the interval graph with the highest frequency as a linkage pattern. However, frequent patterns that are accidentally constructed by noise (pseudo patterns) cause the following problems. If different pseudo pattern labels are attached to the same interval graphs, these interval graphs are considered completely different despite having an identical linkage pattern. This reduces the accuracy of linkage pattern mining.

C. *Extraction of linkage patterns based on closed itemset*

Pseudo patterns tend to occur randomly on the time axis; thus, the probability that the same pseudo pattern will be included in multiple equivalent interval graphs is extremely low. Therefore, it is expected that pseudo patterns can be excluded by extracting label sets that occur commonly in multiple interval graphs. The proposed method extracts clear linkage pattern without the pseudo patterns by closed itemset mining on the obtained interval graphs.

Fig. 2c shows the process of excluding pseudo patterns from interval graphs. Each interval graph is considered a transaction, and each node in the interval graph is considered an item. By applying closed itemset mining to this transaction database, we can extract the maximal node sets (closed itemsets) that are shared in *minsup* or more interval graphs. Finally, the closed itemset with the highest frequency is output as the linkage pattern. Thus, it is possible to extract linkage patterns with greater accuracy as randomly constructed pseudo patterns can be excluded. Fig. 2c illustrates an example of how pseudo patterns n_A, n_B, and n_C are excluded; only the authentic linkage patterns {A, B, C} are extracted.

In this study, we use the fast and exhaustive linear closed itemset miner (LCM) algorithm [22].

V. Experiments

The proposed method was evaluated for extraction accuracy and computational time using artificially created sequential datasets.

A. *Artificial datasets*

Each artificial dataset comprised three sequential data. The sequential data were generated by inserting 10 linkage patterns (embedded linkage patterns) into random sequential data created using uniform random numbers. For this experiment, we created five non-noise artificial datasets (Dataset1–Dataset5) that included no noise within the embedded linkage patterns. Fig. 3 shows a section of each artificial dataset. The formats of linkage patterns embedded in each dataset are as follows. Dataset1 is an artificial dataset wherein equal length frequent patterns were embedded with the same start time across the three sequential data (Fig. 3a). Dataset2 is an artificial dataset wherein equal length frequent patterns were embedded with different start times across the three sequential data (Fig. 3b). Dataset3 is an artificial dataset wherein different length frequent patterns for each of the three sequential data were embedded at the same time (Fig. 3c).

(Revised online publication: 22 July 2016)

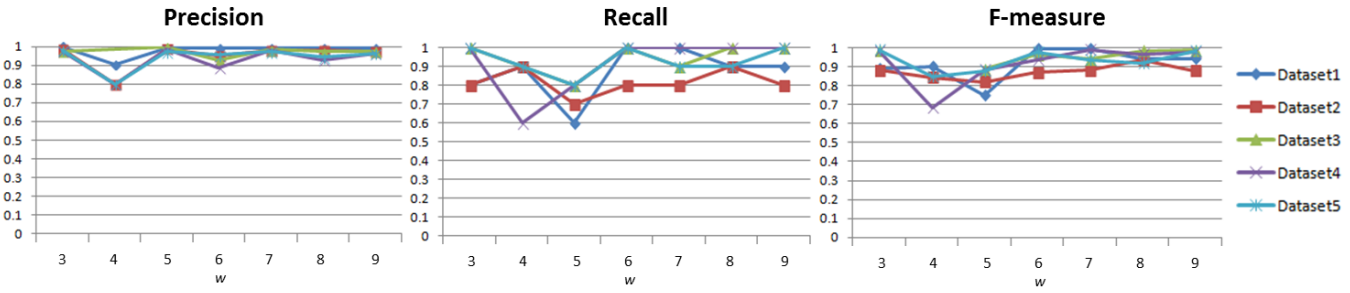

Fig. 4. Extraction accuracies in different *w* for the datasets without noise by the previous method

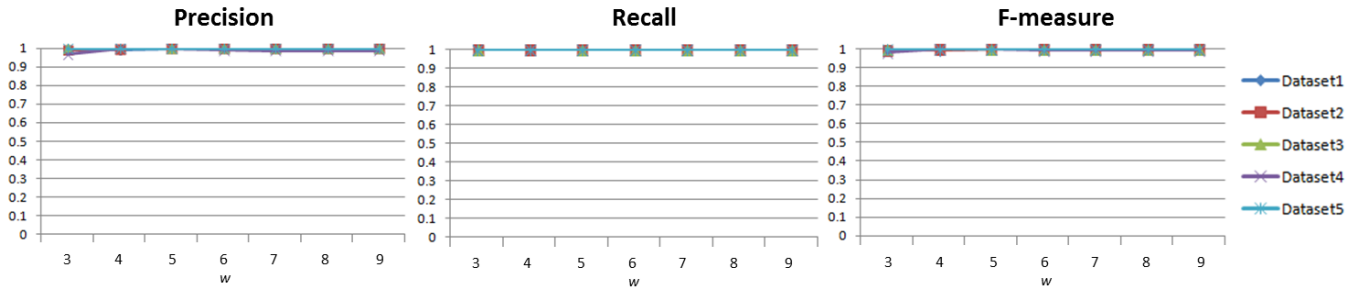

Fig. 5. Extraction accuracies in different *w* for the datasets without noise by the proposed method

Dataset4 is an artificial dataset wherein frequent patterns with different lengths for each of the three sequential data were embedded at different times (Fig. 3d). Dataset5 is an artificial dataset wherein one or two types of frequent patterns were embedded with different lengths and different start times for each of the three sequential data (Fig. 3e).

In addition, five artificial datasets (Dataset1_noise–Dataset5_noise) that included noise in the embedded linkage patterns were created by adding fluctuations to each time point in the linkage patterns. The fluctuations were generated using normal random numbers (SD = 0.01).

B. *Parameter settings*

For frequent pattern extraction, the minimum number of occurrences θ was fixed at 5, and the window widths *w* were set to natural numbers ≥3. For closed itemset mining, *minsup* were set to natural numbers ≥2.

C. *Extraction accuracy of linkage patterns*

The extraction accuracies of the embedded linkage patterns for the previous and proposed methods were compared using the above 10 artificial datasets. *Precision*, *recall*, and *F-measure* were used as evaluation indexes. These indexes were calculated as follows.

$$Precision = CDP/DDP$$
$$Recall = CDP/EDP$$
$$F\text{-}measure = 2 * Precision * Recall/(Precision + Recall)$$

Here, CDP is the number of data points in the correctly detected areas of the embedded linkage patterns, DDP is the number of data points in the areas of the embedded linkage patterns detected by the method, and EDP is the number of data points in the embedded linkage patterns.

D. *Evaluation of computational time*

This experiment was conducted using the five noisy datasets (Dataset1_noise–Dataset5_noise).

The window width *w* significantly affected the computational time required to find frequent patterns [13]. First, we evaluated the computational time for the range of *w* described in Section IV.

In addition, sequential data length may also largely influence the computational time. Therefore, we increased the length by linking each dataset together and measured computational time when modifying up to 10,000 points in increments of 1,000.

VI. Results and Discussion

A. *Extraction accuracy for non-noise datasets*

Figs. 4 and 5 are graphs of precision, recall, and F-measure in different *w* when the previous and proposed methods were applied to the five non-noise datasets. The *minsup* was fixed at 5. In these graphs, the results in the range $3 \leq w \leq 9$ are shown because no frequent patterns were extracted in $w \geq 10$.

As we can see, the previous method shows unstable scores for different *w* values. This is caused by the pseudo patterns randomly formed by noise added to the embedded linkage patterns. In contrast, the proposed method demonstrates 100% extraction accuracy for $w > 4$. This means that the noises included in the interval graphs were suitably excluded by closed itemset mining.

B. *Extraction accuracy for noise datasets*

This experiment was conducted using parameters (*minsup* = 5, $3 \leq w \leq 9$) same as those in the previous section. In the previous method, the accuracy of extracting linkage patterns was 0% for all datasets because only one interval graph was

(Revised online publication: 22 July 2016)

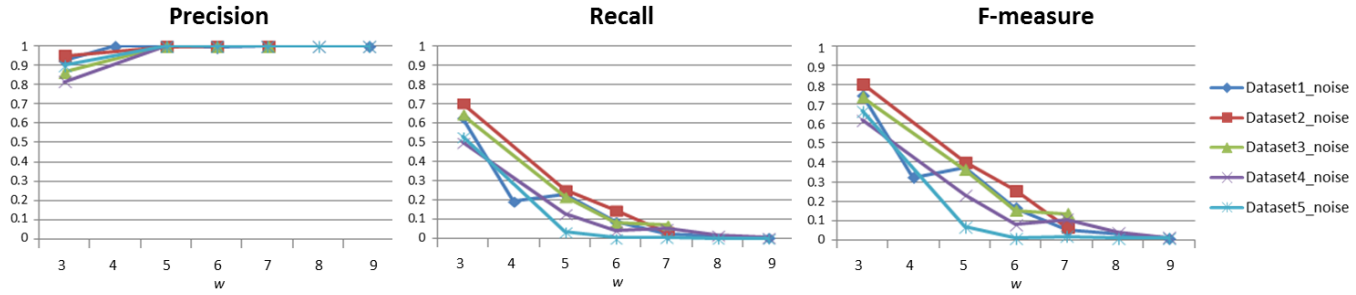

Fig. 6. Extraction accuracies in different *w* for the datasets with noise by the proposed method

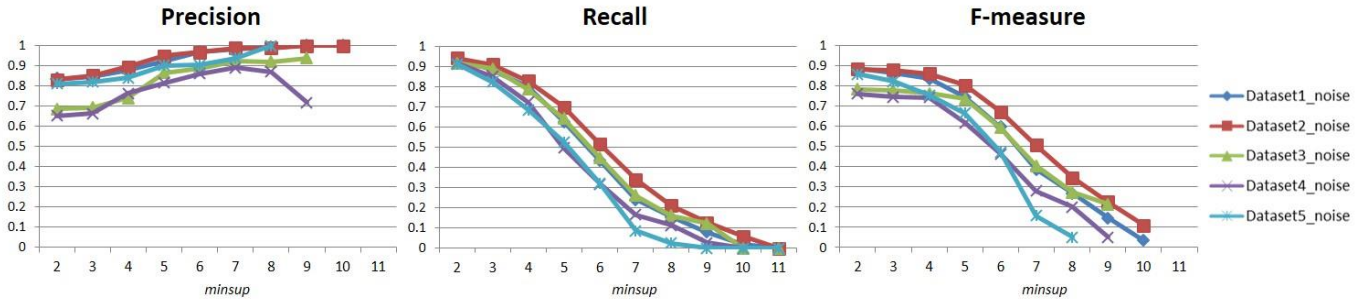

Fig. 7. Extraction accuracies in different *minsup* for the datasets with noise by the proposed method

generated. This is because pseudo patterns exist throughout the sequential data. Thus, only the results of the proposed method are shown in this section. Fig. 6 shows graphs of precision, recall, and F-measure in different *w* for the five datasets with noise (Dataset1_noise–Dataset5_noise). The precision values for all datasets are ≥ 80% for all *w* values. In particular, when $w \geq 5$, the embedded linkage patterns are effectively extracted from all datasets because pseudo patterns are suitably excluded by closed itemset mining. Note that recall tends to decrease as *w* increases. In particular, when $w \geq 5$, recall decreases drastically for all datasets because the number of frequent patterns extracted from each sequence decreases drastically. Therefore, the obtained interval graphs are also reduced drastically. Note that F-measure decreases significantly with the drastic decline of recall values.

In addition, we investigated the impact of *minsup* on the extraction accuracy. Fig. 7 shows graphs of precision, recall, and F-measure in different *minsup*s. In this experiment, the *w* was fixed at 5. The precision tends to increase with increasing *minsup* in all the datasets. In contrast, the recall decreases dramatically with increasing *minsup*. In particular, a rapid decrease in the scores is observed in $minsup \geq 5$. The F-measure is a similar tendency to the recall and especially shows high scores in the range of $2 \leq minsup \leq 4$.

From the above results, we can see that *w* and *minsup* should be fixed at a smaller value to obtain higher extraction accuracy.

C. Impact of window width on computational time

Fig. 8 shows graphs of computational times when *w* was varied. In this experiment, *minsup* was set to 5. In addition to the total computational time required for all steps in the proposed method, these graphs show the computational time for Steps (a), (b), and (c). Note that Step (a) is frequent pattern extraction and labeling, Step (b) is interval graph generation, and Step (c) is linkage pattern extraction based on closed itemset mining. As we can see, the total computational time is strongly affected by the computational time of Step (a) and increases drastically with increasing *w* because Step (a) must check labels in a combinatorial manner to find frequent patterns. On the other hand, the computational times of Steps (b) and (c) are considerably shorter than Step (a) and relatively stable against the increased *w*. This is due to the following reasons. First, Step (c) only detects the overlapped intervals along the time axis and therefore can be executed in linear time for the sequential data length. Furthermore, with regard to Step (c), besides the closed itemset enumeration algorithm LCM being exceptionally fast, the size of the transaction database for interval graphs was small (only tens to hundreds of transactions). Thus, computational time is highly dependent on the time required to extract frequent patterns; however, it is possible to execute within a realistic time by reducing the *w* value.

D. Impact of sequential data length on computational time

Fig. 9 shows graphs of computational time for each step, including the total time when sequential data length was changed. In this experiment, *w* and *minsup* were set to 5. As we can see, Steps (a) and (b) increase linearly with increased sequential data length. However, Step (a) requires more computational time than Step (b) owing to the combinatorial search in frequent pattern extraction. For Step (c), the computational times are considerably less than Steps (a) and (b) although there are major fluctuations related to sequential data length. This is because the size of the transaction database changed depending on the number of extracted interval graphs. From the above, we can see that the computational time of the proposed method increases linearly with increased sequential data length. However, the computational time required to extract frequent patterns constitutes a large proportion of the proposed method's total

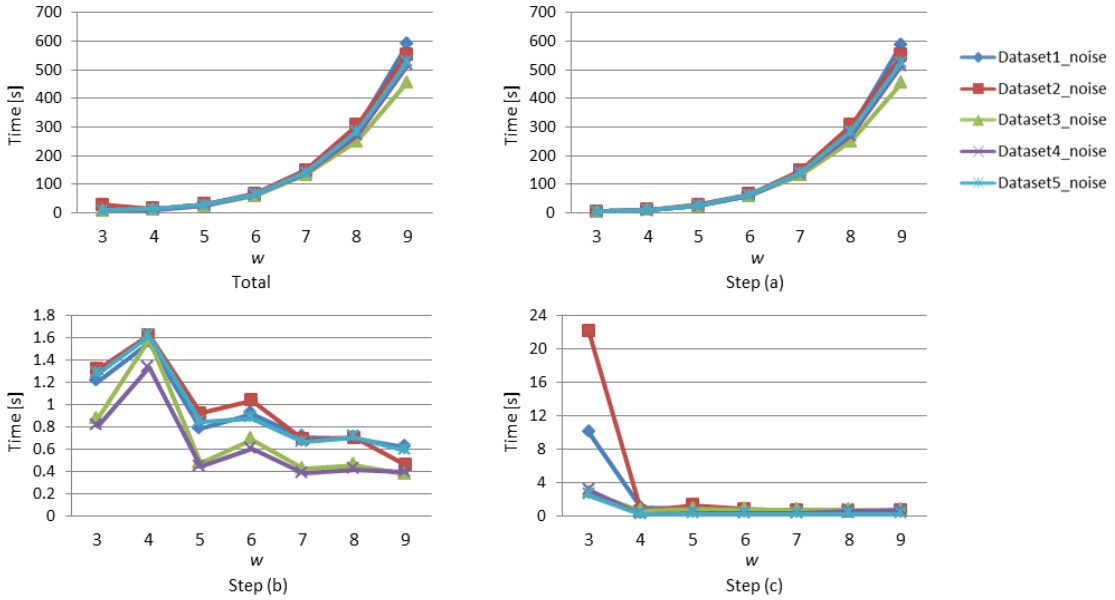

Fig. 8. Computational times in different *w*

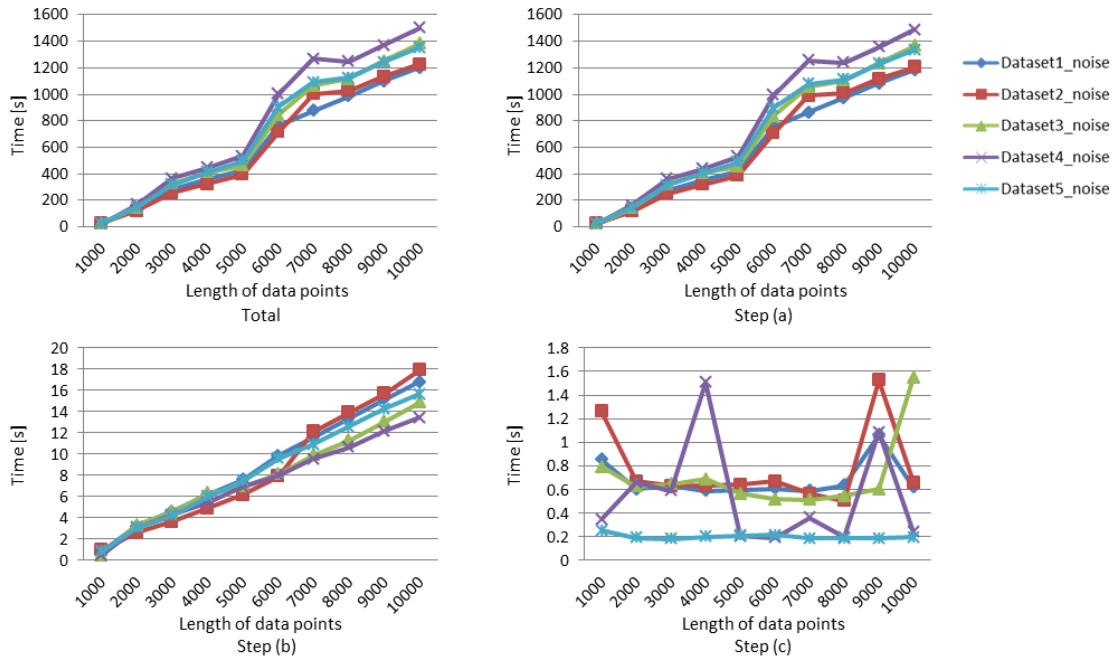

Fig. 9. Computational times in sequential data of different lengths

computational time. Increasing the speed of the frequent pattern mining algorithm will certainly become an issue when applying this method to large-scale real data.

VII. Conclusion

We proposed a new noise-robust linkage pattern mining method based on closed itemset mining. In the proposed method, closed itemset mining is employed to exclude pseudo patterns generated by noise/fluctuations and obtain only frequent and maximal patterns among different interval graphs. In our first experiment, we compared the performance of the previous and proposed methods using artificial datasets. As a result, it was shown that the proposed method can appropriately detect linkage patterns with noise that were not detected by the previous method. Furthermore, we found that *w* and *minsup* should be fixed at a smaller value to obtain higher extraction accuracy. In our second experiment, we measured computational time using five datasets with noise when the window width *w* and sequential data length were varied. As a result, we observed that computational time increases as *w* and sequential data length increase. Furthermore, in the proposed method, the impact of introducing closed itemset mining on computational time is substantially small.

In future, we will address increasing the speed of the frequent pattern mining algorithm. In addition, we will apply the method to large-scale real sequential data that includes noise/fluctuations, such as vital data and crustal movement data. The practical applicability of the proposed method will

(Revised online publication: 22 July 2016)

also be evaluated in terms of extraction accuracy and computational time.

REFERENCES

[1] R. Agrawal and R. Srikant, "Mining sequential patterns," in *1995 Proc. 11th Int. Conf. on Data Engineering*, pp. 3–14.

[2] F. Takchungm, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, 2011, pp. 164–181.

[3] Q. Zhao and S. S. Bhowmick, "Sequential pattern mining: A survey. technical report." *CAIS*, Nanyang Technological University, Singapore, No. 2003118, 2003.

[4] C. I. Ezeife and Y. Lu, "Mining web log sequential patterns with position coded pre-order linked WAP-tree," *Data Mining and Knowledge Discovery*, vol. 10, pp. 5–38, 2005c Springer Science, Business Media. Inc. Manufactured in The Netherlands.

[5] X. Wu, Y. Wu, Y. Wang, and Y. Li, "Privacy-aware market basket data set generation: A feasible approach for inverse frequent set mining," in *2005 Proc. 5th SLAM Int. Conf. on Data Mining*, pp 103–114

[6] A. D. Lattner, A. Miene, U. Visser, and O. Herzog, "Sequential pattern mining for situation and behavior prediction in simulated robotic soccer," *RoboCup 2005: Robot Soccer World Cup IX Lecture Notes in Computer Science*, vol. 4020, pp 118–129, 2006.

[7] R. Sarno, R. D. Dewandono, T. Ahmad, M. F. Naufal, and F. Sinaga, "Hybrid association rule learning and process mining for fraud detection," *IAENG International Journal of Computer Science*, vol.42, no. 2, pp. 59–72, 2015.

[8] M. Karaca, M. Bilgen, A. N. Onus, A. G. Ince, and S. Y. Elmasulu, "Exact tandem repeats analyzer (E-TRA): A new program for DNA sequence mining," *J. Genet.*, vol. 84, pp. 49–54, 2005.

[9] H. Ohtani, T. Kida, T. Uno, and H. Arimura, "Efficient Serial Episode Mining with Minimal Occurrences," in *3rd Int. Conf. on Ubiquitous Information Management and Communication*, 2009.

[10] P. Wen-Chi and L. Zhung-Xun, "Mining sequential patterns across multiple sequence databases," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 1014–1033, 2009.

[11] C. Gong, W. Xindong, and Z. Xingquan, "Mining sequential patterns across time sequences," *New Generation Computing*, vol. 26, pp. 75–96, 2008.

[12] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth." in *2001 Proc. 17th Int. Conf. on Data Engineering*, pp. 215–224.

[13] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1, pp. 259–289, 1997.

[14] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance" in *2007 Proc. ICDE*, pp. 1046–1055.

[15] Y. Sakurai, S. Papadimitriou, and C. Faloutsos, "BRAID: Stream mining through group lag correlations." in *2005 Proc. ACM SIGMOD Conf.*, pp. 599–610.

[16] Y. Zhu and D. Shasha, "StatStream: Statistical monitoring of thousands of data streams in real time" in *2002 Proc. of VLDB*, pp. 358–369.

[17] T. Miura and Y. Okada, "Detection of linkage patterns repeating across multiple sequential data," *Int. J. Computer Applications*, vol. 63, no. 3, pp. 14–17, 2013.

[18] L. Saerom, T. Miura, and Y. Okada, "A new method for improving the performance of linkage pattern mining," in *2014 Proc. of IMECS*, pp. 36–40.

[19] N. Miyoshi, T. Shigezumi, R. Uehara, and O. Watanabe, "Scale free interval graphs," *Theoretical Computer Science*, vol. 410, no. 45, pp. 4588–4600, 2009.

[20] N. Korte and R. H. Mohring, "An incremental linear-time algorithm for recognizing interval graphs," *SIAM J. Computing*, vol. 18, pp. 68–81, 1979.

[21] G. S. Lueker and K. S. Booth, "A linear time algorithm for deciding interval graph isomorphism." *J. ACM*, vol. 26, pp. 183–195, 1979.

[22] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining," in *2005 Proc. of 1st Int. Workshop on Open Source Data Mining*, pp. 77–86.

(Revised online publication: 22 July 2016)

1) Date of modification
2016/07/22

2) Brief description of the changes
Fig. 7 in the above paper was printed mistakenly. The correct figure is reprinted below.
By this revision, there is no change in the conclusion of the paper.

Before revision

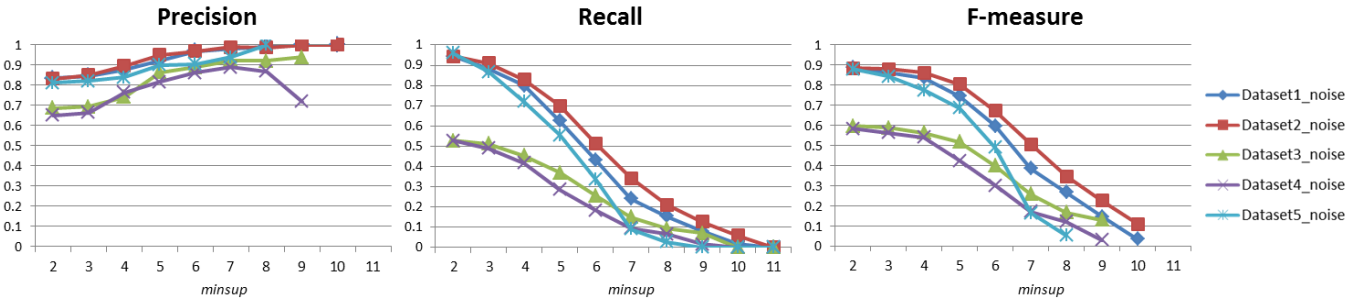

Fig. 7. Extraction accuracies in different *minsup* for the datasets with noise by the proposed method

After revision

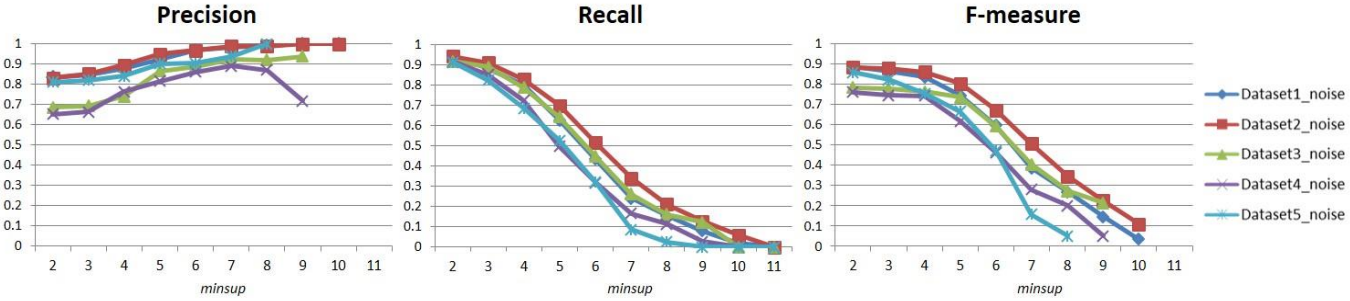

Fig. 7. Extraction accuracies in different *minsup* for the datasets with noise by the proposed method

(Revised online publication: 22 July 2016)